Subject: Re: alternative to execute
Posted by Robert Barnett on Tue, 31 Jan 2006 22:02:45 GMT
View Forum Message <> Reply to Message

Writing out 300 functions would seen like a bit of a drain, but it
might be your only choice if you cannot see any pattern across the
variation of these functions. Presuming there are some patterns to
simplify the problem then I'd probably use object oriented design. I've
been in similar situations before and I've found OO often helps you
implement the patterns.

The way I approached my similar problem was to:

Step 1: Create a base class which has common methods and fields across
all of your functions.

```
pro arithmetic__define, struct
struct = {arithmetic, field1: ptr_new(), field2: ptr_new(), field3:
ptr_new()}
end
```

Step 2: Create class definitions for every function. (Use a perl script
or something to generate the code)

```
pro artimetic1__define, struct
struct = {arithmetic1, INHERITS arithmetic}
end

pro artimetic2__define, struct
struct = {arithmetic2, INHERITS arithmetic}
end

pro artimetic3__define, struct
struct = {arithmetic3, INHERITS arithmetic}
end
```

Step 3: Implement your functions across all classes

```
pro artimetic1::solve, b123, b100, b050, b035, RESULT=result
result = b123 / b100 - *self.field1 * b050 / b035
end

pro artimetic2::solve, b123, b050, b035, RESULT=result
result = b123 - *self.field1 * b050 / b035
end

pro artimetic3::solve, b123, b050, b035, RESULT=result
result = b123 - b050 / *self.field1 * b035
```

```
end
```

The advantage of this style of programming is that you can use
inheritance to tweak functions. Take your function, "arithmetic2".
You could create two tweaked version "arithmetic2a" and
"arithmetic2b".

For example:

```
pro artimetic2::solve, b123, b050, b035, RESULT=result
result = b123 - *self.field1 * self -> apply(b050, b035 )
end

pro artimetic2a__define, struct
struct = {arithmetic2a, INHERITS arithmetic2}
end

pro artimetic2b__define, struct
struct = {arithmetic2b, INHERITS arithmetic2}
end

function artimetic2a::apply, a, b
return, a*b
end

function artimetic2b::apply, a, b
return, a/b
end
```

I've really only skimmed the surface of OO design here. For brevity I
haven't shown the init, cleanup or Get/Set property methods. But,
hopefully it is food for thought.

P.S. You could half the code written here if CREATE_STRUCT accepted the
INHERITS keyword.

Robbie